Lab 11: non-linear models

Statistics for Social Scientists II

Bur, GJM

2024-11-18

1 Non-linear models

In general, linear models are of the form $Y = \beta_0 + \beta_1 X_1 + \epsilon$. So, non-linear models are those where either Y, X, or both are subject to some non-linear transformation. Today, we'll work with quadratic models and logarithmic models.

In these notes, I use population notation for models to reduce notational clutter, so $Y = \beta_0 + \beta_1 X_1 + \epsilon$ in place of the sample estimates $Y = \widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \epsilon$. Nothing about what these notes discuss really touches on statistical inference: it's more about how to interpret these models. So, just plug in sample estimates for these things. Also, in general, notation like y_0, x_0 and y_1, x_1 refer to points some function—the lowercase means "actual values of Y and X"; in particular, don't confuse x_1, some coordinate point, with X_1, where the subscript is there in case we have more than one X. These notes are all about bivariate models for simplicity, though.

2 Power rules

It might be useful to review these before looking at the rest of the document. Most power rules are easy to prove by simply writing out what it represents. Here are some simple ones.

1. $x^p \cdot x^q = x^{p+q}$

   To prove, simply pick two small values for p and q and then multiply all the ys.xou'll write p of them first, then q, but this is obviously x^{p+q}.

2. $(x^p)^q = x^{pq}$

   To prove, once again we simply pick two feasibly-small values and write this out: we multiply $x^p q$ times, which is equivalent to finding x^{pq}.

3. $\frac{x^p}{x^q} = x^{p-q}$

   To prove in the simple case where $p > q > 0$, simply write this out. The number of ys which cancel from the numerator is q; this leaves $p - q$ left over. When we come to non-obvious situations, like those where the exponents are the same or the denominator's is larger, we define them so that rule 3 holds. To wit…

4. $x^0 = 1$

   We do this so that the subtraction rule in 3 works.

5. $x^{-p} = \frac{1}{x^p}$

   Again, we do this so that the subtraction rule in 3 works.

3 Quadratic equations

Quadratic equations are those of the form $y = ax^2 + bx + c$. Curves with positive a point up (convex); curves with negative a point down (concave). We might think that some relationships between sociological variables have this property; many people suggest that age should have a relationship like this with many variables since people tend to be most dependent and least active in social life when they are either children or retired.

3.1 Basic analysis

Let's look at an example using the CPS. We can do this by making a squared term in Stata or with factor notation. Let's look first at the hand method.

```
gen sq_age = age^2
scatter imphrs age
```

It looks like we have a strong case for part-time workers. Let's just examine them; for a fuller analysis, you could just have a fully-interacted model, but let's focus on the simple case. Note that while the pattern is *visually* totally obvious, there is basically *no* linear component to the relationship: `corr imphrs age`. *It is important to consider the possibility that you have such patterns!*

```
gen pt = imphrs<35
keep if pt == 1
reg imphrs age sq_age
```

How do we interpret this model? Is there any sense in assigning meaning to the main effect? If not, what else might we do with it? (Below, I give answer to those questions).

One useful thing to do to get a handle on the model is to calculate predictions for the observations in our data-set. **Try doing this by hand (use `coeflegend`) and then check your work with Stata's automatic prediction calculator, shown below**.

Now, let's look at the predictions as a function of age along with a scatterplot. *Note that in what follows, the `sort` option is essential to make the line look right.* As the help file (`help line`) notes, `Be sure that the data are in the order of the x variable, or specify line's sort option. If you do neither, you will get something that looks like the scribblings of a child.`[1]

```
predict yhat, xb
scatter imphrs age  ///
```

[1]In fact, when I accidentally ran a chunk of code without it, Stata froze and my work since my last checkpoint was lost: be careful asking Stata to do ambitious things with big data-sets!

```
    || line yhat age, sort lc(black)
```

Let's now look at the factor syntax approach. Note that age is essentially just being interacted with itself, so we write the model in that way.

```
reg imphrs c.age##c.age
qui sum age
```

We can run our F-test as well since our linear term is nested in the full model. I'll show it by hand since we have been doing that a lot lately; practice makes perfect.

```
reg hrsimp age
local SSM_r = e(mss)
reg wage4 c.age##c.age
local SSM_f = e(mss)
local mean_sq_e = e(rmse)^2
local SSM_incr = `SSM_f'-`SSM_r'
local df_m_diff = 1
local mean_sq_m = `SSM_incr'/`df_m_diff'
local F = `mean_sq_m'/`mean_sq_e'
di `F'
test c.age#c.age
```

And we can examine our predictions. Here, I show how to inductively get the increments in the `margins` command. I just tell Stata to get what feels offhandedly like me to a good x-increment ($\frac{s}{4}$) and then pull the observed min/max, rounded to an integer, from the summary.

```
qui sum age
local incr = round(r(sd)/4, 1)
local min = round(r(min), 1)
local max = round(r(max), 1)
margins, at(age=(`min'(`incr')`max'))
marginsplot, ytitle("Predicted hours")
```

Try to calculate the predicted values for $age = 40$ by hand (use `coeflegend` if you like!), and then do so with `margins`: `margins, at(age=40 sq_age=1600)`. Compare this to a conditional sum for people who are 40: `sum imphrs if age == 40`.

Note that we can also track the effect of age *in our model* across values of age. This equation will be that of a line—but because this is the *effect*, that means that we have a non-linear model (because the effect changes over time. Note that the line also has components both above and below the x-axis: the effect is positive initially, then later negative.

```
qui sum age
local incr = round(r(sd)/4, 1)
local min = round(r(min), 1)
local max = round(r(max), 1)
margins, dydx(age) at(age=(`min'(`incr')`max'))
marginsplot, ytitle("Estimated effect of age")
```

3.2 Tangent lines

Note that the main effect—the slope β_1—*is hard to interpret, and the easiest interpretation is that it is the tangent line at* $X_1 = 0$.[2]

So, one more-meaningful thing that we can do is find the tangent line to our prediction equation at various points to get a linear approximation the neighborhood; we'll denote that point x_0 and the predicted value there $\hat{y}_0$.

In general, the slope of the tangent line for $y = ax^2 + bx + c$ is given by $2ax + b$, or for us, $2\beta_2 X_1 + \beta_1$. To find the tangent at some point x_0, just swap in x_0 for X_1. The full equation of our tangent line is then $y = (2\beta_2 x_0 + \beta_1)x + b$. Note that here b does not mean the same b as for the quadratic equation; it's just a generic intercept (I didn't want to introduce a ton of new notation).

To get the intercept b of the tangent line, we want it to pass through the predicted value $\hat{y}_{x_0}$, the value on our prediction equation that results from plugging in x_0. The easiest way to do this is put the prediction equation into point-slope form, which is very useful to have handy for these notes.

Recall the point-slope equation of a line. It takes the line $y = ax + b$ and rewrites it as a distance from a point on the line $(x_0, y_0) : (y - y_0) = a(x - x_0)$.

$$\begin{aligned} y_0 &= ax_0 + b \\ y &= ax + b \\ y_1 - y_0 &= a(x - x_0) \\ y_1 &= a(x - x_0) + y_0 \end{aligned}$$

So, our tangent line starting at x_0 and passing through $\hat{y}_0$ is

$$y - \hat{y}_0 = (2\beta_2 x_0 + \beta_1)(x - x_0)$$

It's usually easier to plug in at this point and rearrange, but if you want to directly solve for b, just rearrange the equation above to isolate y, and b is just all the stuff not multiplied by a variable x: $b = \hat{y}_0 - (2\beta_2 x_0 + \beta_1)x_0$.

Here's an example for $x_0 = 30$, first by hand and then in Stata. The quadratic equation I get from Stata's regression is $y = 10.5318 + 0.5872x - 0.0065x^2$. So, the slope of the tangent line anywhere is $-0.013x + 0.5872$ and the slope of the tangent line here is $-0.013(30) + 0.5872 = 0.1972$. The predicted value $\hat{y}_0$ at this point is 22.2978

[2] You can fix this to some extent by centering, as usual, but I don't discuss that here.

$$\begin{aligned}
y - \hat{y}_0 &= (2\beta_2 x_0 + \beta_1)(x - x_0) \\
y - 22.2978 &= 0.1972(x - 30) \\
y &= 0.1972(x - 30) + 22.2978 \\
y &= 0.1972x + 16.3818
\end{aligned}$$

Here it is in Stata. Note that this must be run after a `reg` command.

```
local yhat_0 = _b[age]*30 + _b[c.age#c.age]*900 + _b[_cons]
local a = 2*_b[c.age#c.age]*30 + _b[age]
local b = `yhat_0' - `a'*30
capture drop tangent
gen tangent = `a'*age + `b'

line yhat age, sort || line tangent age, sort
```

Try all of that with $x_0 = 50$.

3.3 The vertex (change of direction point) of the quadratic function

We can also find the point where the curve bends by finding where the tangent line's slope equals 0; simply solve for $2\beta_2 x + \beta_1 = 0$, getting $x = -\frac{\beta_1}{2\beta_2 x}$. **Find where the total age effect on imputed hours starts becoming negative and plot a vertical line with `xline()`.**[3]

3.4 Local changes

Finally, we can also use our linear approximation to the function to find approximate predicted values for some point $\hat{y}_1$ at x_1, where x_1 is "in the neighborhood of" x_0. What this tells us is that if the tangent line is a good *local* approximation to the curve, we can use the slope at a particular place to make predictions. We just modify our point slope formula.

$$\begin{aligned}
y_1 - y_0 &\approx 2(\beta_2 x_0 + \beta_1) \cdot (x_1 - x_0) && \text{For small changes in } x \\
y_1 &\approx 2(\beta_2 x_0 + \beta_1) \cdot (x_1 - x_0) + y_0
\end{aligned}$$

So, for example, if $x_0 = 30$, then $2(\beta_2 30 + \beta_1) \cdot 1 + \hat{y}_0$ gives a good approximation for $\hat{y}_{31}$. Working by hand, we get...

[3]Something like this after the relevant `reg` command:

```
local argmin = -_b[age]/(2*_b[c.age#c.age])
line yhat age, sort xline(`argmin', lc(black))
```

$$\begin{aligned}
y_1 - 22.2978 &\approx 0.1972(x_1 - 30) \\
y_1 &\approx 0.1972(31 - 30) + 22.2978 \\
y_1 &\approx 0.1972 + 22.2978 \\
y_1 &\approx 22.495
\end{aligned}$$

```
qui reg imphrs c.age##c.age
local yhat_x0 = _b[age]*30 + _b[c.age#c.age]*900 + _b[_cons]
local yhat_x1 = `yhat_x0' + 2*_b[c.age#c.age]*30 + _b[age]
di `yhat_x1'
margins, at(age=(30 31))
```

Time-permitting: **Try all of that with** $x_0 = 50$.

4 Log-lin models

4.1 Log properties

Logarithms are kind of like roots in that they can take a set of numbers which is in some way exponentially distributed and return one which is more compressed—perhaps Normal or perhaps uniform. There are many exponential distributions and many relatively-compressed distributions, but this is the general effect. This effect can happen in multiple senses.

First, let's think about how logarithms and roots work generally. In these cases, I use x to mean the quantity that we don't know but want to find; however, you'll usually see x represent the argument to the function in other contexts, including below. But here it highlights a key difference.

$$\begin{aligned}
&\text{root problems}: x^p = k && p \text{ and } k \text{ known} \\
&\text{root solutions}: \sqrt[p]{x^p} = \sqrt[p]{k} \to x = \sqrt[p]{k} \\
&\text{log problems}: b^x = k && b \text{ and } k \text{ known} \\
&\text{log solutions}: \log_b b^x = \log_b k \to x = \log_b k
\end{aligned}$$

So, both functions in some sense "invert" exponential types of expressions.

Logarithms can also be thought of in this way: logarithms turn products into sums, divisions into differences, and exponentiation into multiplication. To see this, let's note the most famous log property, that $\log_b xy = \log_b x + \log_b y$. To see this proof, merely note that for *some* p, q, no matter how weird, x and y can be expressed as b^p and b^q, respectively.

$$\begin{aligned}
\log_b xy &= \log_b b^p b^q \\
&= \log_b b^{p+q} \\
&= p + q \\
&= \log_b x + \log_b y
\end{aligned}$$

Now, let's examine the power property.

$$\begin{aligned}
b^p &= x \\
(b^p)^q &= x^q \\
b^{pq} &= x^q \\
\log_b b^{pq} &= \log_b x^q \\
pq &= \log_b x^q \\
q \log_b x &= \log_b x^q
\end{aligned}$$

The division property follows immediately from these two ; just write $\log_b \frac{y}{x} = \log_b yx^{-1} = \log_b x + \log_b y^{-1} = \log_b x - \log_b y$.

4.2 Logarithms for compressing univariate distributions

So, taking logarithms of numbers is one way to "undo" exponentiation. Now, statistical distributions may or may not follow some exact exponential distribution. However, taking the logarithm of such a distribution is valid either way. You can think about it like this: taking the logarithm with base b of a variable Y would produce a nice Normal or uniform distribution if $Y \sim b^X$ and $X \sim U(a, b)$ or $X \sim N(\mu, \sigma^2)$. If the distribution is *not* some exact exponential function, that is OK; we're basically saying that it is close enough, and if that's true, taking a logarithm will have a comparable effect (if that's a false assumption, there's still no harm really). By the way, why take a logarithm instead of a root? No strong reason, but note that b^x quickly becomes much larger than x^b, so taking a logarithm is a stronger transform or "undo" of the original series (because it presumes our model is the former function).

Let's look at probably the most common example in social science, the ubiquitous "log wages"

```
use ./data/cps2019_bigextract, clear
hist wage4
gen log_wages = log(wage4)
hist log_wages
foreach var of varlist wage4 log_wages {
    sum `var'
    gen z_`var' = (`var' - r(mean))/r(sd)
}
sum z*, d

twoway (hist z_wage4, percent color(red%55)) ///
    (hist z_log, percent color(green%55) ///
    legend(order(1 "original" 2 "logarithm")))
```

So, we can make our univariate distributions look nicer and more like things we recognize.

Notice in the above that there was an X, though. So, what if we have access to information on X? Well, our model above implies that $\log_b Y = \log_b b^X \to \log_b Y = X$, exactly or more realistically as a linear function of X with some noise. In the exact case, this basically means that our Y and X are really the same conceptual variable.

4.3 Log-lin models proper

But there are cases where it is reasonable to think that this is a *statistical* relationship between two variables, and the standard way to model it is what is known as a "log-lin" model. Even if our model is a bit more complicated, $Y = e^{\beta_0 + \beta_1 X + \epsilon}$, note that this still implies that logging both sides produces a model we know how to deal with well: $\log_b Y = \beta_0 + \beta_1 X + \epsilon$. *In particular, using the natural logarithm, i.e.* $\log_e = \ln$, *produces very easily interpretable results.* Stata's default logarithm when we write `log(X)` is the natural log, by the way. So when you see log from here on out, think "natural logarithm".

How can we make sense of this model? Let's look at an example.

```
clear all
use ./data/nhanes1719-extract.dta

hist bmxwt
scatter bmxwt bmxht || lfit bmxwt bmxht, lc(black)
```

The pattern here again looks to be of the form $Y =$ Some exponential function involving X. We have already explored one possibility, that X_1 should be treated as the argument to a quadratic function. But, as noted above, one other possibility is that maybe X_1 should be treated as the argument to an exponentiation function—in simple terms, what if X_1 is not *being* squared but is the *exponent* of some base (again, almost always e for reasons discussed below). This leads to a model of the form $Y = e^{\beta_0 + \beta_1 X_1 + \epsilon}$, where we allow for the exponent to be a linear function of X_1 rather than just X_1. And conveniently, this can be easily expressed by logging both sides as $\ln Y = \beta_0 + \beta_1 X_1 + \epsilon$. **Let's estimate such a model. Find the logarithm of weight using gen `ln_wt = log(bmxwt)` and then run a regression. Examine the results.**

4.4 Interpreting log-lin models

Now, we have a couple of ways to interpret our slope. The simpler one is an approximation, though one that works very well here: the coefficient on a log-lin model can be interpreted as "a unit change in X predicts a $\beta_1 \cdot 100$ percent change in Y on average" *when* β_1 *is near* 0. Notice, by the way, that before with quadratic equations our approximations were valid when changes in X were close to zero; now, our approximations are valid when β_1 is close to zero. **Now interpret the coefficient**.

We can also manually and exactly calculate the *ratio* change in a predicted Y for a given change in X, usually just a unit change, with exponents.

$$\begin{aligned}\frac{\mathbb{E}[Y|X_1 = x+1]}{\mathbb{E}[Y|X_1 = x]} &= \frac{e^{\beta_0+\beta_1(x+1)}}{e^{\beta_0+\beta_1 x}} \\ &= e^{\beta_1} && \text{See power laws above}\end{aligned}$$

This implies that at any particular value of X, the ratio change is given by β_1; or, in other words, we can multiply e^{β_1} by the predicted value at any X to get the predicted value for a unit change in X.

Let's try this out. Using the regression you carried out above, find the actual predicted change for a unit change in X. Hint: rerun the regression with `, coeflegend` and then invoke `exp(·)` on one of the results. Was our approximation pretty good?

Now, let's plot our results. There is a particularly satisfying way to see that we've **maybe** done something correct here. First, obtain your predictions for the logged Y after running the regression: `predict ln_yhat, xb`. Then, exponentiate the predictions to get them on the scale of the *original* Y: `gen yhat = exp(ln_yhat)`. Then, plot them with the original scatterplot: `scatter bmxwt bmxht || line yhat bmxht, sort lc(black)`. The `sort` option is essential to get the line to plot correctly; the linecolor (`lc`) option is just useful for certain color schemes.

Here is a full comparison of the linear vs. logged models:

```
scatter bmxwt bmxht, ///
    legend(order(2 "Log fit" 3 "Linear fit") ///
        pos(11) ring(0) region(lstyle(solid))) ///
    ytitle("Weight (kg)") ///
    || line yhat bmxht, sort lc(black) ///
    || lfit bmxwt bmxht, lc(black) lp(-)
```

We could also get our original predictions back with `margins`, predictably.

```
reg ln_wt bmxht
qui sum bmxht
local incr = round(r(sd)/4, 1)
local min = round(r(min), 1)
local max = round(r(max), 1)
margins, at(bmxht=(`min'(`incr')`max')) expression(exp(predict(xb)))
marginsplot, ytitle("Predicted weight given height")
```

Let's calculate a couple of predicted values.

```
margins, expression(exp(predict(xb))) at(bmxht=(99 100 149 150))
```

See if our two interpretations hold for the two unit changes implied—roughly: our slope tells us that a unit change in X leads to a $100\beta_1$ percent increase in Y; precisely, $\widehat{y}_{X=x} = e^{\widehat{\beta}_1} \cdot \widehat{y}_{X=x-1}$.

4.5 Model comparison

Finally, is our model actually any better? We can compare the two quickly, without looking at a ton of regression output.

```
qui reg bmxwt bmxht
di e(r2_a)
qui reg ln_wt bmxht
di e(r2_a)
```

However, note that our outcome is not actually the same, so this approach isn't really doing anything meaningful. Per the textbook, we could try regressing our original outcome on our exponentiated predicted values from the logged model. Here, that actually gives a *worse* fit!

```
reg bmxwt yhat
```

The short answer for "what to do" is to pick a model that seems theoretically more appropriate. There is also recourse to looking for which model has better or worse residuals.

```
reg bmxwt yhat
predict linear_errors, residuals
hist linear_
reg ln_wt bmxht
predict log_errors, residuals
hist log_errors
qui qnorm linear, name(linerr)
qui qnorm log, name(logerr)
gr combine linerr logerr
```

5 Why the $\ln$ approximations work + lin-log and log-log models

Here is a simple proof for why that you should be able to follow, although the fact that $e^{\beta_1} \approx 1 + \beta_1$ for β_1 close to 0 requires a bit of calculus in the appendix.

$$\begin{aligned}
\ln Y &= \beta_0 + \beta_1 X_1 + \epsilon && \text{Our model's slope-intercept form}\\
y_1 - y_0 &= \beta_1(x_1 - x_0) && \text{Point-slope form for any line}\\
\ln y_1 - \ln y_0 &= \beta_1(x_1 - x_0) && \text{Point-slope form applied to our model for } x_1 = x_0 + 1\\
\ln y_1 - \ln y_0 &= \beta_1 && \text{Assume } x_1 - x_0 \text{ to be a unit change}\\
\ln\left\{\frac{Y_1}{Y_0}\right\} &= \beta_1 && \text{Log properties}\\
\frac{Y_1}{Y_0} &= e^{\beta_1} && \text{Exponentiate both sides}\\
e^{\beta_1} &\approx 1 + \beta_1 && \text{for } \beta_1 \text{ close to 0}\\
\frac{Y_1}{Y_0} - 1 &\approx \beta_1\\
\frac{Y_1 - Y_0}{Y_0} &\approx \beta_1
\end{aligned}$$

By a similar logic, we can figure out how to interpret lin-log models for small changes in X. The short takeaway is that for lin-log models, a *one percent* change in β_1 leads to a $0.01\beta_1$ change in Y. We use here the fact that the point-slope formula is *approximately* true for functions that aren't linear in X if we swap in the tangent slope at some point $x_0 : y_1 - y_0 \approx \text{tangent slope}(x_1 - x_0)$. We also use the fact that the tangent slope to a function $k \ln X$ at any point x_0 is $\frac{k}{x_0}$ (see appendix for the calculus).

$$\begin{aligned}
Y &= \beta_0 + \beta_1 \log X_1 + \epsilon\\
y_1 - y_0 &\approx \text{tangent slope}(x_1 - x_0) && \text{True in the neighborhood of } x_0\\
y_1 - y_0 &\approx \frac{\beta_1}{x_0}(x_1 - x_0)
\end{aligned}$$

And then one interesting small change about x_0 is to simply increase it by a factor of 1.01, leaving...

$$\begin{aligned}
y_1 - y_0 &\approx \frac{\beta_1}{x_0}(1.01x_0 - x_0)\\
y_1 - y_0 &\approx \frac{\beta_1}{x_0}(.01x_0)\\
y_1 - y_0 &\approx .01\beta_1
\end{aligned}$$

Finally, for log-log models, we just combine the two interpretations. For small βs, a one percent change in X gives approximately a β_1 percent change in Y. This is sometimes called the *elasticity* of Y with respect to X.

6 Appendix

6.1 A simple proof of the derivative of $ax^2 + bx$

Properly proving this requires the binomial theorem; I have notes on this if you want.

However, there is a very quick way to prove the power rule for the simple case for x^2 and x (and to take the derivative of a sum, we intuitively take the sume of the two derivatives). Let dx mean "the smallest possible change in x we can measure" and dy mean "some small, unknown change in y".

$$\begin{aligned} y &= ax^2 \\ y + dy &= (ax + adx)^2 \\ y + dy &= (ax)^2 + 2ax(dx) + (adx)^2 \\ dy &= 2ax(dx) + (adx)^2 \end{aligned}$$

If dx is as small as it gets, then squaring it makes it basically negligible. So then we have...

$$\begin{aligned} dy &= 2ax(dx) + 0 \\ \frac{dy}{dx} &= 2ax \end{aligned}$$

Then, the rule for $y = bx$ is even simpler.

$$\begin{aligned} y &= bx \\ y + dy &= bx + b(dx) \\ dy &= b(dx) \\ \frac{dy}{dx} &= b \end{aligned}$$

Simple proofs in calculus along these lines can be found in Silvanus P. Thompson's cult classic *Calculus Made Easy*. These proofs might seem like cheating, and they aren't rigorous, but as U. Louisiana-Lafayette math professor Arturo Magidin points out in this excellent *StackExchange* post, this is how Leibniz conceived of these things, and he was basically always correct—the rough intuition proceeded the rigorous proofs with limits.

6.2 Derivative of e^x

There's kind of a cute way to get this one with only basic calculus (bizarrely, many proofs online involve properties that you would only know with 20th century calculus; seems kind of dumb to me since it is *Euler*'s number!. We'll actually find the derivative of the logarithm of $\ln(x)$ first.

$$
\begin{aligned}
y &= \ln(x) \\
\frac{dy}{dx} &= \lim_{h\to 0} \frac{\ln(x+h)-\ln(x)}{h} \\
&= \lim_{h\to 0} \ln\left\{\frac{x+h}{x}\right\}^{\frac{1}{h}} \\
&= \lim_{h\to 0} \ln\left\{1+\frac{h}{x}\right\}^{\frac{1}{h}} && \text{Log product and power rule} \\
&= \lim_{h\to 0} \ln\left\{1+\frac{h}{x}\right\}^{\frac{x}{h}\cdot\frac{1}{x}} \\
&= \lim_{h\to 0} \ln e^{\frac{1}{x}} \\
&= \frac{1}{x}
\end{aligned}
$$

Now, let's play a little fast and loose and treat $\frac{dy}{dx}$ as a fraction—after all, basically all of the Calc. I properties we know to be true with complete certainty were gotten this way by Leibniz and Newton, Leibniz in particular futzing around with infinitesimals and doing a lot of handwaving.

Then, $\frac{dx}{dy}$ is simply x, and since $y = \ln x, x = e^y$. Or, if we write it in the standard way, if $y = e^x$, $\frac{dy}{dx} = e^x$.

6.3 Taylor approximations

Recall that the logic of the Taylor series is as follows.

1. We want to find a polynomial approximation $T(x)$ to a non-linear function $f(x)$.
2. If our function T has the same kth derivative as f for large k at some value $x = a$, they should be fairly similar around a.
3. So, we find the zeroth, first, second, third ... kth derivatives of f at $x = a$. Since we evaluate the derivative at a specific point, this just results in a number. The zeroth derivative of a function is just the function itself.
4. So, we want the kth derivative of T to result in this number. How can we do this so that this is true for *every* derivative? We can write the polynomial like this: $\frac{f^{(k)}(a)(x-a)^k}{j!}$ will cause the power rule's operation on the $(x-a)^{(k)}$ term to cancel out the factorial in the denominator, leaving only $f^{(k)}(a)$ behind.
5. What about terms of smaller degree? They will go to zero since the kth derivative of T causes a term like $\frac{f^{(k-1)}(a)(x-a)^{(k-1)}}{(k-1)!}$ to be multiplied by zero at some point.
6. What about terms of larger degree? A term like $\frac{f^{(j+1)}(a)(x-a)^{j+1}}{(j+1)!}$ will result in, for the kth derivative e.g., $f^{(k)}(a)(x-a)^1$, but now if we plug in $x = a$, this term evaporates. So, $T^{(k)}(a) = f^{(k)}(a)$.

Now, let's find the Taylor expansion for $f(\theta) = e^\theta$ about $a = 0$ (using θ in place of X here since it plays a different role in our notation).

$$
\begin{aligned}
T^{(k)}[f(\theta)] &= \sum_{j=0}^{k} f^{(j)}(a)\frac{(\theta-a)^j}{j!} \\
T^{(k)}(0) &= \sum_{j=0}^{k} e^{\theta}\Big|_{\theta=0}\frac{\theta^j}{j!} \\
&= \sum_{j=0}^{k} 1\frac{\theta^j}{j!} \\
&= 1+\theta+\frac{\theta^2}{2!}++\frac{\theta^3}{3!}
\end{aligned}
$$

And we can just clip the terms after the first two for a pretty good approximation.

Here is the proof of lin-log models with calculus.

$$
\begin{aligned}
Y &= \beta_0 + \beta_1 \log X_1 + \epsilon \\
\frac{\partial y}{\partial x} &= \beta_1 \frac{1}{X_1} \\
y_1 - y_0 &\approx \frac{\partial y}{\partial x}(x_1 - x_0) \\
y_1 - y_0 &\approx \frac{\beta_1}{x_0}(x_1 - x_0)
\end{aligned}
$$

And then one interesting small change about x_0 is to simply increase it by a factor of 1.01, leaving...

$$
\begin{aligned}
y_1 - y_0 &\approx \frac{\beta_1}{x_0}(.01x_0) \\
y_1 - y_0 &\approx .01\beta_1
\end{aligned}
$$